Titre du projet : Phishing
Nom de l'équipe : croissant
Membre : Élisa Correia-Martins (1880313)

Prétraitement des attributs (Feature Design)

La première étape du prétraitement des attributs a été de les standardiser. Pour cela, la classe *StandardScaler* a été utilisée. On utilise d'abord la méthode *fit_transform* sur les données d'entraînement, puis la méthode *transform* sur les données de validation et les données de test.

Ensuite, on utilise la méthode *SelectKBest* du module *feature_selection* de la librairie de *sklearn* pour sélectionner les features à garder pour le modèle. Les features qui ont obtenu un score en dessous de 1 sont retirés. Les features retirés sont les suivants : *nb_space*, *nb_redirection*, *char_repeat*, *iframe*, *onmouseover* et *right_clic*.

Méthodologie

Répartition des données : Selon plusieurs sources, il est recommandé d'utiliser 20-30% des données disponibles pour effectuer la validation. Cependant, après plusieurs tests, il a été conclu que pour le modèle, il est plus bénéfique de descendre ce nombre à 15% car on arrive à une meilleure précision.

Techniques pour gérer le déséquilibre entre les classes :

Puisqu'il n'y a pas de déséquilibre important entre les classes (4012 « phishing » et 3989 « legitimate ») aucune technique a été appliquée pour les balancer. Cependant, pour éviter le débalancement lors de la séparation des données d'entraînement en données de validation, on utilise le paramètre stratify lors de l'appel à la méthode *train_test_split* qui fait en sorte que la proportion des classes avant la séparation soit maintenue pour chaque séparation.

Stratégie de régularisation :

La stratégie de régularisation utilisée est celle du Early Stopping, qui arrête l'apprentissage du modèle lorsque les performances diminuent, dans ce cas-ci lorsque la valeur du validation loss cesse de diminuer pendant 30 epochs.

Réglage des hyperparamètres :

Nombre de « hidden layers » et de neurones : Le nombre de « hidden layers » a été chosi afin de maximiser la précision tout en évitant le surapprentissage. On a d'abord commencé avec une seule couche cachée et graduellement augmenté le nombre. À partir de trois couches cachées, on peut apercevoir qu'il y a du surapprentissage, donc on est resté à deux couches. Le nombre de neurones par couche a aussi été choisi en testant graduellement dans la même logique que pour le nombre de hidden layers.

Learning rate : Le learning rate a été choisi de manière à ce que le modèle apprenne suffisamment tout en évitant que la divergence se produise trop tôt.

Nombre d'epochs: Le nombre d'epochs a été fixé à 2000. Cependant, le modèle arrête son entraînement lorsque la valeur du « validation_loss » n'a pas diminué depuis 50 itérations avec l'ajout d'une fonction de rappel « EarlyStopping ». Ceci permet de maximiser le temps d'apprentissage en évitant d'aller trop

loin, car si le nombre d'epochs est trop élevé, la valeur du validation loss commence à remonter, ce qui est signe de surapprentissage.

Batch size : La sélection du batch size s'est faite par essais et erreur. On a testé les valeurs de 16, 32, 64, 128, 256 et 512. Les différences en performance n'étaient pas flagrantes. La valeur de 128 a finalement été choisie, car elle semblait donner des performances légèrement meilleures.

Résultats

Note: les courbes oranges représentent les valeurs de validation et les courbes bleues, les valeurs de l'entraînement

Performances obtenues du modèle :

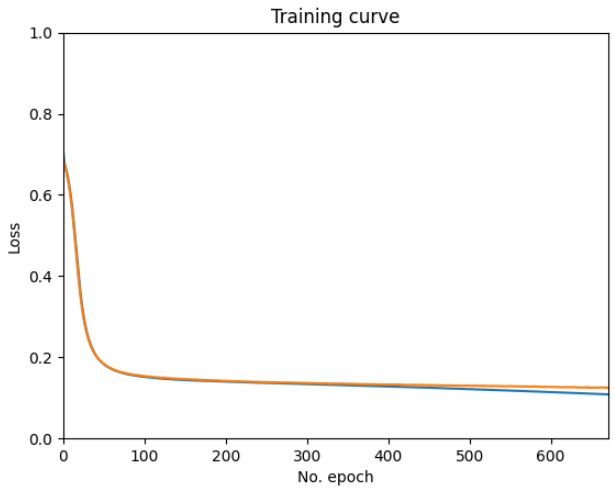

Figure 1: Loss du modèle final

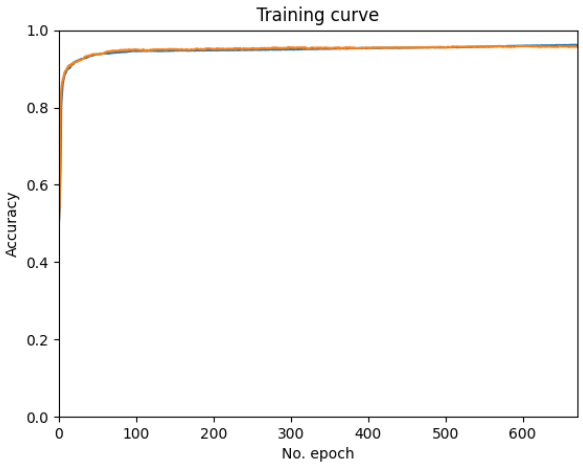

Figure 2: Accuracy du modèle final

loss = 0.1087
accuracy = 0.9615
val_loss = 0.1250
val_accuracy = 0.9567

Prétraitement :

Standardisation des données :

Sans standardiser les données au préalable, on remarque une baisse importante dans la précision. On obtient une valeur de précision finale de 0.7036. On remarque que le modèle est incapable d'apprendre, les courbes sont relativement plates et bruitées. Ceci est en grande partie dû au fait que les features n'ont pas la même échelle.

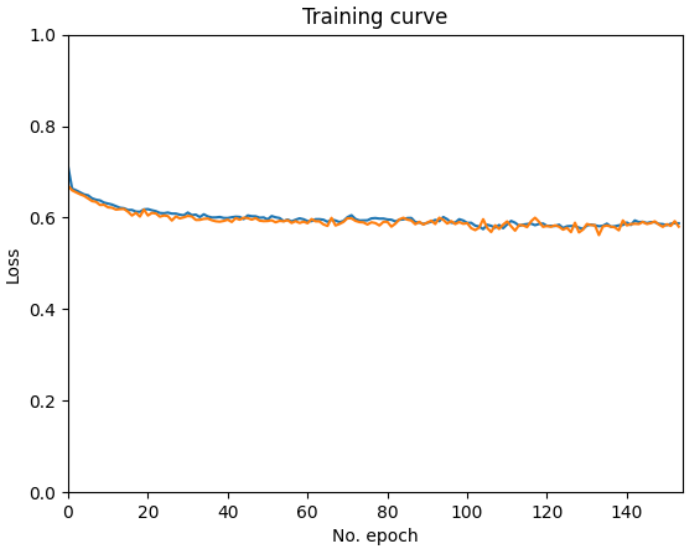

Figure 3: Loss sans standardisation des données

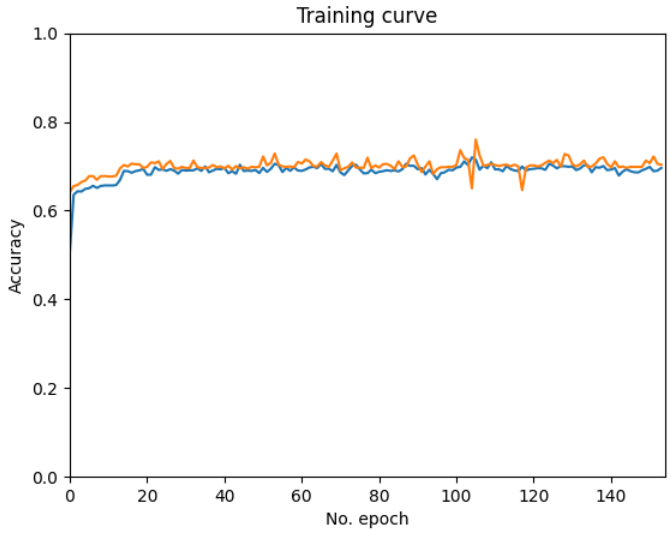

Figure 4: Accuracy sans standardisation des données

Abandon de features selon les scores obtenus :

Cette étape du processus aurait pu être omise. On remarque qu'on obtient plus ou moins les mêmes performances. Il est possible que cette méthode ait un impact plus important si le seuil choisi était différent. On choisit de retirer les features dont le score obtenu selon la méthode SelectKBest est plus petite que 1, ce qui retire au total 6 features.

Régularisation :

L'utilisation du Early Stopping arrête le modèle avant qu'il ne diminue trop en termes de performance. Sans Early Stopping, l'apprentissage continu au-delà de ce qui est attendu et éventuellement régresse en performances. On remarque que la courbe du validation loss diminue pour ensuite augmenter. On veut arrêter l'apprentissage avant que la courbe se remet à monter. La courbe du validation accuracy stagne pendant que la courbe accuracy augmente pour atteindre 1.0. C'est signe que l'apprentissage a duré trop longtemps.

Note : Le nombre d'epochs a été augmenté pour mieux visualiser son effet.

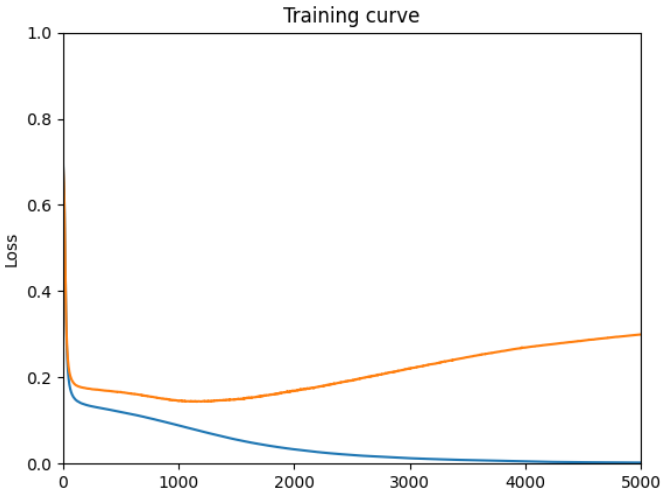

Figure 5: Loss sans Early Stopping

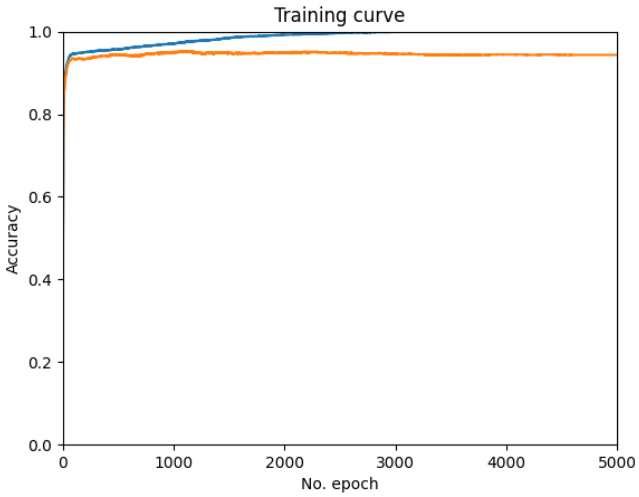

Figure 6: Accuracy sans Early Stopping

Hyperparamètres :

Note: Présentation des hyperparamètres qui ont le plus gros impact sur les performances

Nombre de *hidden layers* :

Lorsqu'on ajoute plus que trois couches cachées, on peut apercevoir des signes de surapprentissage sur le graphique des précisions. Une courbe de type brisée est un signe qu'il y a surapprentissage :

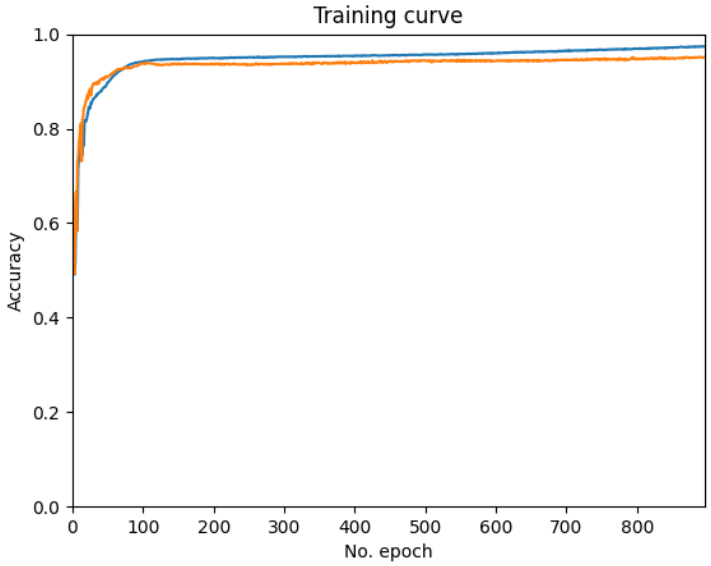

Figure 7: Ajout d'une couche cachée supplémentaire de 5 neurones

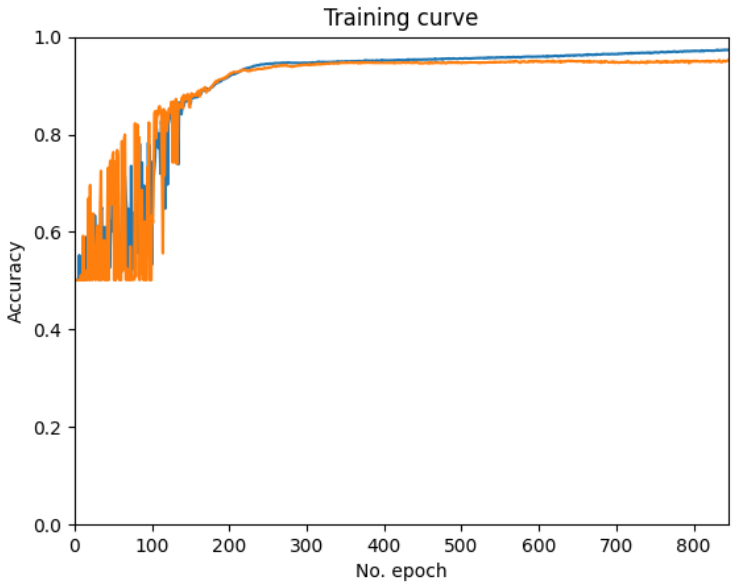

Figure 8: Ajout de deux couches cachées supplémentaires de 5 neurones chacune

Learning rate :

Un *learning rate* trop élevé fait diverger les courbes très rapidement. Avec un learning rate de 0.1, on remarque que la valeur du *validation loss* cesse de diminuer après environ 200 epochs

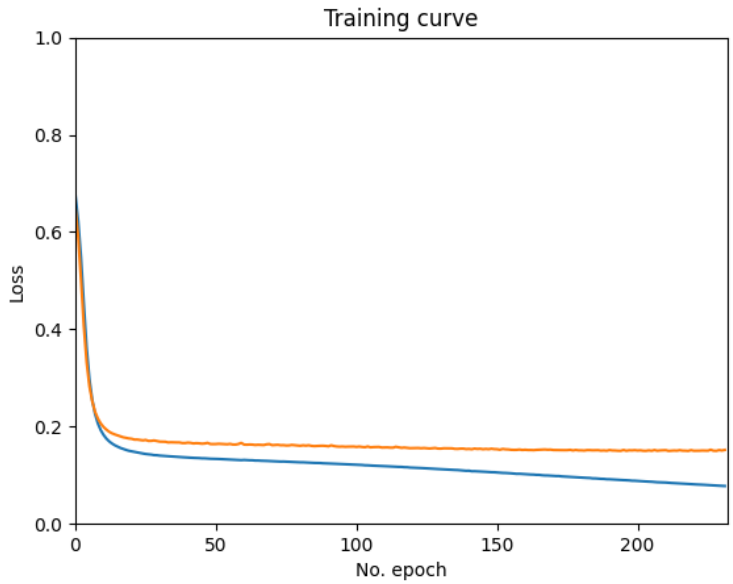

Figure 9: Learning rate de 0.1

Un *learning rate* trop bas ne permet pas d'atteindre une bonne précision en maintenant le niveau maximum d'epochs de 2000. Il est cependant possible d'obtenir des performances similaires en augmentant le nombre d'epochs.

Avec un learning rate de 0.001, on obtient un validation accuracy de 0.8826 avec 2000 epochs :

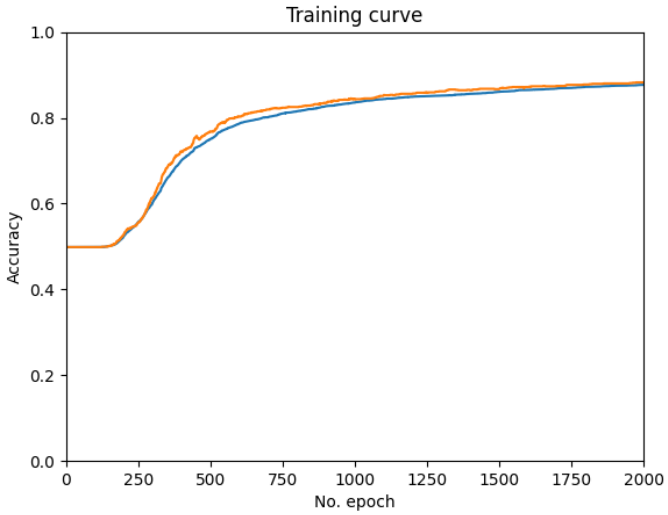

Figure 10: Accuracy avec un learning rate de 0.001

Discussion

Avantages :

Le modèle offre de bonnes performances en relativement peu de temps d'apprentissage. Il permet de retirer certains features qui ne représente que du bruit et alors qui ne contribuent pas positivement à l'apprentissage. Il permet de gérer le débalancement des classes dans les ensembles de données d'entraînement et de validation. Il utilise aussi une technique de régularisation pour limiter le surapprentissage.

Inconvénients :

Le principal inconvénient est la performance. Le modèle n'arrive pas à dépasser le baseline. Une architecture différente ou un meilleur réglage des hyperparamètres aurait pu augmenter la performance du modèle. De plus, le modèle d'implémente pas de «cross-validation» qui offrirait une estimation plus précise sur les performances du modèle. Une technique qui aurait pu être utilisée est celle du « k-fold Cross-Validation ».